Accepted in IEEE INDISCON 2026

ElasticLB: Seamless Auto-Scaling and Load Balancing for Cloud-Native 5G Core Networks

Vraj Patel, Harsh Nema, Rishabh Jain, and Samaresh Bera, *Senior Member, IEEE*
Department of Computer Science and Engineering
Indian Institute of Technology Jammu, Jammu and Kashmir, India - 181221
Email: {2022ucs0100, 2022ucs0091, 2022ucs0105}@iitjammu.ac.in, s.bera.1989@ieee.org

***Abstract*—In this paper, we present `ElasticLB`, an elastic and lightweight framework designed for load balancing and auto-scaling of Access and Mobility Management Function (AMF) instances in cloud-native 5G core networks. The proposed framework dynamically distributes control-plane traffic among multiple AMF instances based on their real-time capacity and adjusts the cluster size in response to fluctuating network demand. During traffic surges, ElasticLB proactively provisions additional AMF instances to maintain signaling performance and avoid congestion. Conversely, during periods of low traffic, the framework improves resource efficiency by identifying under-utilized AMF instances and consolidating their sessions onto other active instances. We consider a seamless session-migration mechanism that enables the transfer of active sessions between AMF instances without service interruption, thereby preserving the quality of the user experience. We evaluate the performance of `ElasticLB` on a 5G testbed under stochastic user equipment arrival and departure patterns. Experimental results demonstrate that `ElasticLB` significantly reduces the number of active AMF instances compared to static deployment strategies, while maintaining stable control-plane latency. Additionally, the framework achieves lower CPU and memory utilization, making it well-suited for resource-constrained private 5G deployments.**

***Index Terms*—5G core networks, Access and Mobility Management Function (AMF), Auto-scaling, Load balancing, Kubernetes, Open5GS, Seamless migration**

I. Introduction

The transition to 5G standalone networks represents a fundamental shift in mobile core architecture, replacing monolithic hardware appliances with a Service-Based Architecture (SBA) composed of virtualized and cloud-native network functions. This transformation is driven by the need to support a wide range of heterogeneous and demanding use cases, including enhanced Mobile Broadband (eMBB), ultra-Reliable and Low-Latency Communications (uRLLC), and massive Machine-Type Communications (mMTC) [1]. With the rapid growth in the number of connected devices, projected to reach billions of IoT endpoints, the volume of control-plane signaling in 5G core networks is expected to increase by several orders of magnitude compared to 4G LTE systems [2], [3].

The Access and Mobility Management Function (AMF) acts as the primary entry point for such signaling, handling critical tasks such as user registration, authentication, and mobility management. As a result, ensuring the scalability and reliability of AMF deployments is essential for maintaining the overall stability and performance of the 5G core network. However, provisioning AMF clusters to handle peak traffic demand results in significant resource inefficiency and increased operational expenditure, particularly due to the bursty and diurnal characteristics of mobile network traffic. Static deployment strategies require operators to maintain excess capacity to accommodate infrequent "signaling storms", leading to prolonged periods of underutilized CPU and memory resources. Therefore, improving resource utilization in the 5G core is a critical requirement for achieving sustainability and cost optimization in modern telecommunications infrastructures.

While Kubernetes [4] and other cloud-native orchestration frameworks provide mechanisms for horizontal scaling, such as the horizontal pod autoscaler [5], they are primarily designed for stateless web applications. Applying these generic frameworks to 5G network functions, such as AMFs, introduces unique challenges due to the stateful nature of control-plane signaling. Existing studies [6]–[9], have investigated dynamic scaling to address traffic surges, with a primary focus on proactive scale-up strategies to minimize latency. However, these approaches often overlook the complexities associated with safe scale-down operations. Arbitrary termination of an active AMF instance can disconnect all associated user-equipment (UE) and base-station (gNB) pairs, leading to service disruptions. Moreover, such termination can trigger a surge of re-registration requests that may further destabilize the network. Motivated by these limitations, we pose the following research questions:

- How to auto-scale (up and down) AMFs dynamically based on real-time control traffic?
- How to seamlessly migrate live control sessions from one AMF to the other due to scale-down and/or load-balancing across active AMFs?
- What would be a viable approach for practical cloud-native deployments of 5G core networks, specifically, AMFs?

To address the above questions, we propose `ElasticLB`, a framework for AMF load balancing and auto-scaling in a cloud-native 5G network. `ElasticLB` adopts a lightweight proxy-based architecture that abstracts the AMF cluster from the Radio Access Network (RAN), enabling centralized and intelligent traffic management. Unlike generic auto-scaling mechanisms, `ElasticLB` maintains awareness of active connections associated with each AMF instance. Leveraging this

information, it dynamically scales the AMF cluster using two dedicated algorithms: one for scale-up and another for scale-down. During scale-down operations, `ElasticLB` identifies underutilized AMF instances and performs seamless migration of their active N2 associations to other instances prior to termination. This migration process ensures uninterrupted service continuity, allowing UEs to remain connected while the system consolidates resources. We evaluate `ElasticLB` on a 5G laboratory testbed, with the core network deployed using Open5GS [10]. User association requests are modeled using a General Independent (GI) arrival process and Markovian (M) service times, corresponding to a GI/M/c queuing model [11]. Experimental results demonstrate that `ElasticLB` effectively balances resource utilization and control-plane latency under dynamic traffic conditions in 5G networks.

The rest of the paper is organized as follows. Section II presents the state-of-the-art load balancing and auto-scaling approaches in 5G network. Section III presents the system architecture of `ElasticLB`. We discuss the implementation details in Section IV. The experiment results are presented and discussed in Section V. Finally, Section VI concludes the paper.

II. Related Work

Load balancing and auto-scaling of virtualized network functions have received significant attention in mobile core networks [6]–[8], [12]–[15]. Early foundational work by Buyakar et al. [7] developed prototypes for distributing control-plane traffic among multiple AMFs, demonstrating the importance of load balancing in reducing registration and signaling latency. Dev et al. [9] introduced a load balancing framework, called ScaleLB, which combines proactive and reactive scale-up techniques with round-robin load balancing. ScaleLB activates additional AMF instances when traffic exceeds a threshold, improving control-plane performance under load spikes. However, ScaleLB focuses primarily on scale-up and does not address how to safely scale down or migrate active sessions when demand decreases. As a result, resource consumption may remain high during low-traffic periods, leading to operational inefficiency and higher costs. Recent works have also explored containerized network function deployments and auto-scaling orchestration in Kubernetes environments. Chowman et al. [13] proposed AMF scaling and load balancing framework and reported simulation results. However, such a solution may not be applicable for real-time services due to high time complexity for determining optimal solution. Studies on cloud-native 5G core deployments [16] have shown that containerization enables fine-grained resource control and rapid scaling. However, the existing works mostly focused on infrastructure-level scaling without addressing stateful migration of active sessions which may lead to service disruption during migration process.

Alongside these efforts, machine learning-based traffic prediction techniques have been proposed to anticipate network demand and enable proactive resource allocation. Alawe et al. [17] investigated traffic prediction models for mobile core networks and demonstrated their utilities in enabling more responsive and efficient scaling policies. While such approaches reduce latency by forecasting demand, they introduced additional training and computational overhead. Moreover, such approaches may not be cope up with sudden traffic surge in the presence of diverse 5G applications.

In summary, while prior work advances load balancing and scaling in 4G and 5G systems, important challenges that remain unaddressed are as follows: a) graceful descaling without service disruption; b) transparent migration of active sessions; c) realistic arrival and departure process of user association and de-association requests; and d) unified scale-up and scale-down behavior. This work considers these insights, particularly focusing on utilization-aware descaling, explicit session migration, and comprehensive evaluation under GI/M/c traffic models.

III. System Design and Proposed Approach

A. *ElasticLB Architecture*

The architecture of `ElasticLB` is centered around a lightweight and in-path proxy that mediates all N2 control-plane signaling between the RAN and the 5G core's AMF cluster, as depicted in Figure 1. By acting as a logical intermediary, the system abstracts the underlying AMF instances from the gNBs, which enables centralized connection management and dynamic scaling without requiring modifications to any standard network functions. This design is tailored for a cloud-native 5G standalone architecture, where AMFs are deployed as containerized services. Our primary objective is to enhance resource efficiency by dynamically adapting the number of active AMFs to match real-time traffic fluctuations, thereby balancing the trade-off between signaling latency and operational cost.

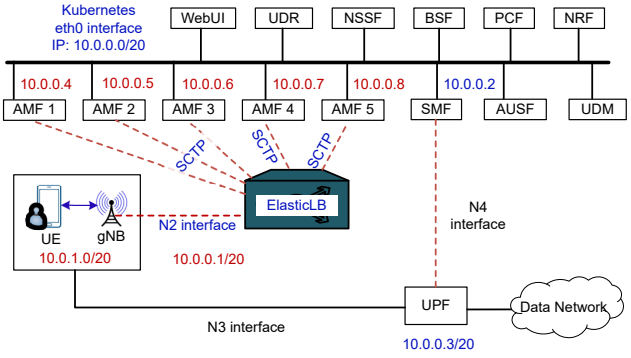

Fig. 1. `ElasticLB` architecture with 5G core network functions

B. *Proposed Approach: Dynamic Scaling of AMFs*

ElasticLB implements a dual policy for managing the AMF cluster size. A proactive scale-up policy prevents performance degradation during traffic surges, while a utilization-driven scale-down policy consolidates resources during idle periods.

Scale-Up: The scale-up logic is triggered synchronously upon each new incoming UE-gNB connection attempt. It is

governed by the condition $L \geq \lfloor C \cdot (N - H) \rfloor$, where L is the total number of active connections, C is the capacity in terms of number of connections that an AMF can handle. N is the number of active AMFs, and H is a configurable headroom factor to maintain system's stability in scale-up and scale-down. The proposed scale-up algorithm is presented in Algorithm 1. If the total load surpasses the predefined load threshold and the cluster has not reached its maximum size (refer to Step 3 in Algorithm 1), a new AMF instance is deployed via a Kubernetes API call. This preemptive action ensures that capacity is added before existing AMFs become saturated, thus, preserving low control-plane latency for most users.

Algorithm 1 Proactive Scale-Up on New Connection Arrival

Inputs: Current active AMFs $\mathcal{A}$, total connections L, per-AMF capacity C, headroom factor H, maximum AMFs $N_{\max}$;
Output: Updated set of active AMFs

1: $N \leftarrow |\mathcal{A}|$
2: $T_{\text{up}} \leftarrow \lfloor C \cdot (N - H) \rfloor$
3: **if** $L \geq T_{\text{up}}$ **and** $N < N_{\max}$ **then**
4: **Select** an inactive AMF k
5: **if** k exists **then**
6: `KubeDeploy(amf-`k`-deployment, replicas=1)`
7: $\mathcal{A} = \mathcal{A} \cup \{k\}$

Scale-Down with Migration: The scale-down process runs periodically to identify and decommission under-utilized resources, which is detailed in Algorithm 2. At fixed intervals, the system calculates the utilization for each active AMF. The system's behavior is tuned using the headroom parameter H, from which the scale-down threshold is derived as $T_D = (1 - H)/6$. If an AMF's utilization falls below a descaling threshold, T_D, it is marked as a candidate for removal. The cornerstone of the scale-down mechanism is the stateless session migration procedure. We follow the 'make-before-break' policy to ensure uninterrupted N2 association and control-plane services. A key aspect of our design is the selection of a suitable target AMF, which must have sufficient capacity to absorb the migrated sessions without becoming overloaded. We define the following upper threshold: $T_U = (1-H)/2$ for choosing a target AMF. Therefore, the selected target AMF must have a utilization that is above T_D but below T_U to prevent cascading migrations and to maintain system stability. Once all active sessions associated with an AMF selected for scale-down are migrated, the AMF is terminated to minimize the operational cost.

IV. Implementation on 5G Testbed

`ElasticLB` is implemented as a user-space SCTP proxy integrated into a Kubernetes-based 5G core testbed. All network functions run as containerized services, which allows the platform to take advantage of standard cloud-native mechanisms for deployment, monitoring, and life-cycle management. The proxy mediates all N2 signaling between the gNBs and the AMF cluster and exposes a stable endpoint to the RAN. Whereas the underlying AMF replicas scale in and out transparently, as depicted earlier in Figure 1.

Algorithm 2 Utilization-Driven Scale-Down and Stateless Migration

Inputs: List of active AMFs $\mathcal{A}$, Capacity C, Descaling Threshold T_D, Target Upper Threshold T_U;
Output: Updated set of active AMFs

1: **if** $|\mathcal{A}| \leq 1$ **then return** ▷ Cannot scale down
2: **for all** AMF $i \in \mathcal{A}$ **do**
3: $U_i \leftarrow$ `Act_Conn`$_i/C$
4: **if** $U_i \leq T_D$ **then**
5: **Mark** i as candidate for scale-down
6: **Select** target AMF $j \in \mathcal{A}, j \neq i$ such that:
7: $T_D <$ (`Act_Conn`$_j/C) \leq T_U$
8: **if** Target j found **then**
9: **for all** connection $conn \in i$ **do**
10: $Socket_{new} \leftarrow$ **ConnectTo**(j)
11: `UpdateForwardTable`$(conn, Socket_{new})$
12: `GracefulShutdown`$(Socket_{old})$
13: `Act_Conn`$_j \leftarrow$ `Act_Conn`$_j + 1$
14: **TerminatePod**(i) ▷ Kubernetes API call
15: **return** $\mathcal{A} \leftarrow \mathcal{A} \setminus \{i\}$ ▷ AMF scaled-down

A. Software Stack and Deployment

The 5G core is realized using Open5GS [10]. Each core network function is deployed as an independent Kubernetes deployment in a dedicated `open5gs` namespace. AMF instances are replicated across up to five deployments, each with a single pod replica, and are addressed via cluster-internal virtual IPs. The cluster runs on a single Ubuntu virtual machine equipped with 32 CPU cores and 128 GB of RAM, hosted on a high-end server platform. The RAN side is emulated with my5G-RANTester [18], which provides gNB and UE simulation conforming to 3GPP procedures. The my5G-RANTester pods run in a separate namespace and connects to the proxy at a fixed N2 endpoint. Therefore, `ElasticLB` behaves as a standard AMF, which makes the deployment of 5G core network independent of any modifications to the RAN.

B. ElasticLB Proxy and Control Components

The ElasticLB proxy runs as a standalone process. It listens on a configurable IP address and SCTP port for incoming N2 associations from gNBs. For each accepted gNB connection, the proxy creates dedicated forwarding threads that handle bidirectional message transfer between the gNB and an assigned AMF. Internally, ElasticLB maintains a record of AMF descriptors that store the AMF identifier, IP address, current connection count, and an active flag. This record is used by Algorithms 1 and 2 for load-balancing and scaling operations. We use the round-robin approach for load balancing.

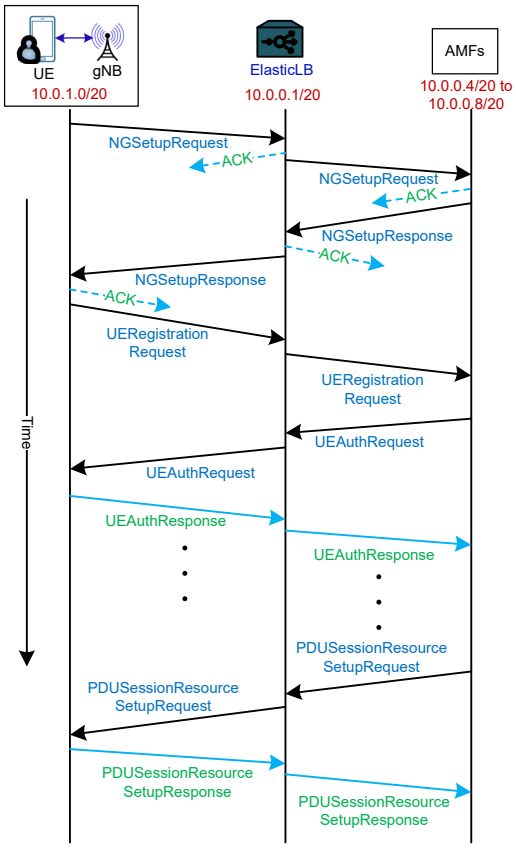

Fig. 2. Message passing for association between UE-gNB and AMFs through `ElasticLB`.

C. Monitoring and Metrics Collection

ElasticLB produces detailed logs for both correctness validation and performance analysis. At startup, the proxy creates a timestamped log directory and initializes three main log streams: a `run-time event log`, a `per-association latency log`, and a `migration log`. For each gNB association, we record the time at which the N2 connection is accepted and the time at which the AMF forwarding path is successfully established. Thus, we measure the latency associated with gNB-AMF connection setup. Resource utilization statistics are collected via the Kubernetes metrics server. A lightweight Python script periodically invokes `kubectl top pod` for the AMF pods every 5-6 seconds and stores CPU usage. Combining the resource utilization records with the AMF activation timeline from ElasticLB logs, we measure how scaling decisions impact resource utilization over time. This instrumentation allows the evaluation to correlate changes in the number of active AMFs with both control-plane latency and resource consumption. Figure 2 shows the abstraction of association procedure between UE-gNB pair and AMFs through `ElasticLB`. Whereas Figure 3 shows the de-association procedure.

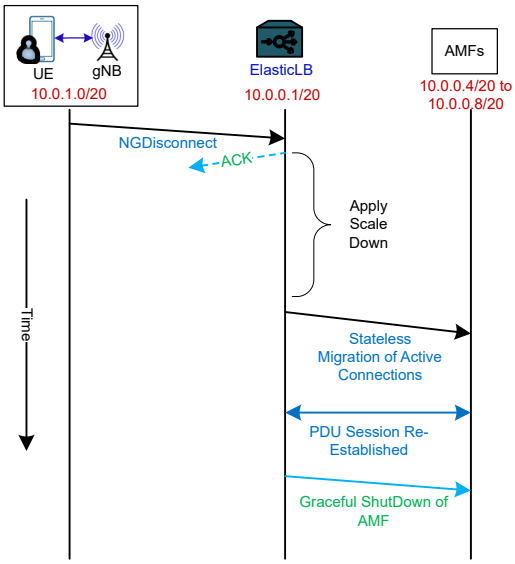

Fig. 3. Message passing for de-association between UE-gNB and AMFs through `ElasticLB`.

D. Traffic Model and Trace Generation

The offered load is constructed in two phases that together approximate a GI/M/c regime [11]: a scripted uniform arrival phase for UE attachments and a trace-driven departure phase for UE session termination. This separation allows the attachment burst and the subsequent decay of load to be controlled independently while preserving a clear mapping to analytical queueing behavior. The arrival and departure configuration are summarized in Table I.

TABLE I
TRAFFIC PARAMETERS

Parameter	Value
Number of UEs (N_{UE})	200
Inter-Arrival model	i.i.d. uniform
Inter-Arrival model (trace generation)	Exponential ($\lambda_{\text{arr}} = 0.5$ s^{-1})
Session duration model	Exponential ($\lambda_{\text{dur}} = 0.05$ s^{-1})
Mean session duration	$1/\lambda_{\text{dur}} = 20$ s

V. Results and Discussion

To show the effectiveness of the proposed scheme, `ElasticLB`, we compare its performance with `Static` approach, in which all AMFs are always active and user association requests are assigned in round-robin fashion to the AMFs. We present the results on the number of active AMFs, control-plane latency, migration statistics, and impact of the headroom on the control-plane latency.

A. Number of Active AMFs

Figure 4 shows the number of active AMFs over different time-interval for `ElasticLB` with different headroom values and `Static` schemes. Due to the scale-up and scale-down

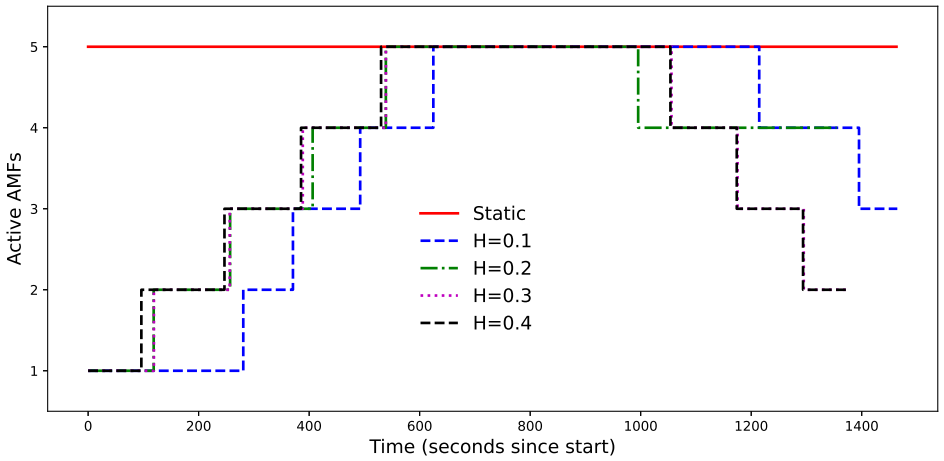

Fig. 4. Auto-scaling of AMFs using ElasticLB compared to Static approach.

features integrated with `ElasticLB`, the number of AMFs are activated based on the actual number of user association/disassociation in the network. Therefore, it is observed that `ElasticLB` gracefully descales, reducing to fewer active AMFs as the load decreases, unlike `Static` approach in which all five AMFs are always active. This also results in reduction of CPU and RAM utilization by 40% compared to `Static` approach.

B. Control-Plane Latency

Figure 5 depicts average connection latency (time from gNB connection to successful AMF assignment) as a function of time. Both `ElasticLB` and `Static` approaches maintain comparable connection latency throughout the experiment. Importantly, `ElasticLB` does not introduce additional latency during scale-down and migration events, which is one of the primary objectives. This is because the proxy layer handles migration transparently without disrupting gNB associations. This validates the effectiveness of stateless migration of `ElasticLB`.

C. Migration Statistics and Impact of Headroom

Table II summarizes the migration events observed during a 200-UE experiment. Scale-down events occur gradually during the detachment phase as utilization crosses thresholds. Migration latencies are in the range of 1–100 ms (largely dependent on the number of sessions to migrate and network conditions). Moreover, UE sessions are not interrupted during migration, confirming the effectiveness of the stateless migration design.

Table III evaluates the sensitivity to the headroom parameter H for the 200-UE experiment. Higher headroom values result in earlier scale-up (more AMFs activated to handle the same load) and reduced latencies due to more available AMFs. Lower headroom enables more aggressive packing, but risks latency spikes under sudden traffic surges.

In summary, the experimental results demonstrate several key strengths of the proposed approach as follows.

TABLE II
MIGRATION STATISTICS (200 UEs, $H = 0.3$)

Metric	**Value**
Total number of scale-down events	3
Total connections migrated	4
Average migration latency (per event)	0.88 ms
Max migration latency (per event)	2.05 ms

TABLE III
SENSITIVITY TO HEADROOM (H) FOR 200-UE EXPERIMENT

H Values	**p95 Latency (ms)**
0.1	1877.79
0.2	1794.55
0.3	2.29

- **Resource Efficiency:** By incorporating scale-down and migration, the system reduces CPU and memory consumption during low-traffic periods compared to static deployments, directly translating to operational cost savings.
- **Competitive Latency:** Connection latencies remain stable across all approaches for most cases, indicating that scaling and migration do introduce control-plane delays for a small fraction of users.
- **Practical Deployability:** The use of Kubernetes-native scaling (pod replicas) and an in-path proxy simplifies deployment and does not require modifications to gNBs, AMFs, or other core network functions. Thus, the proposed solution can easily be integrated into any state-of-the-art cloud-native 5G core deployments.

VI. CONCLUSION

This paper presented ElasticLB, an elastic and lightweight framework for load balancing and auto-scaling of AMF instances in cloud-native 5G core networks. By combining proactive scale-up, utilization-driven scale-down, and trans-

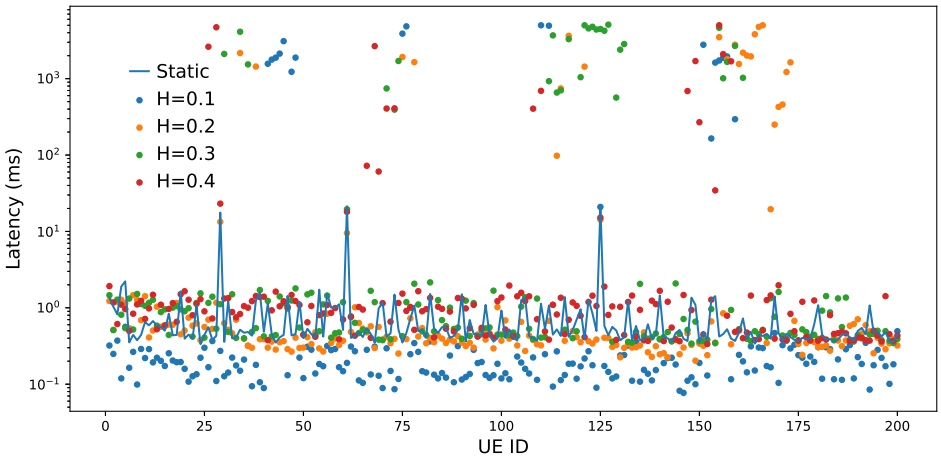

Fig. 5. Distribution of control-plane connection latency for individual UEs under static least-connections and proactive scaling with different headroom values.

parent stateless migration, the system achieves significant resource savings while maintaining stable control-plane latency and service continuity. Experimental evaluation under GI/M/c traffic models demonstrates 40–50% CPU reduction during low-traffic periods compared to static deployments, with zero session interruptions during migration. We plan to integrate AI/ML-based traffic prediction to enable anticipatory scaling as a future extension of this work. Consequently, we plan to use deep reinforcement learning-based approaches for traffic prediction and integrate it with the proposed auto-scaling and load balancing approach. Furthermore, we plan to consider a stateful migration of AMFs that may be one of the key requirements of certain use-case scenarios in 5G and beyond networks.

References

[1] A. Kaloxylos, "A Survey and an Analysis of Network Slicing in 5G Networks," *IEEE Commun. Stand. Mag.*, vol. 2, no. 1, pp. 60–65, Mar. 2018.

[2] "You need a robust signaling solution in 5G too!" Accessed on Jul. 2024, Tech. Rep. [Online]. Available: https://www.ericsson.com/en/blog/2019/10/you-need-a-robust-signaling-solution-in-5g-too

[3] V.-G. Nguyen, K.-J. Grinnemo, J. Taheri, and A. Brunstrom, "Adaptive and Latency-aware Load Balancing for Control Plane Traffic in the 4G/5G Core," in *Joint European Conference on Networks and Communications & 6G Summit*, Jun. 2021, pp. 365–370.

[4] "Production-Grade Container Orchestration," Accessed on Dec. 2025, Tech. Rep. [Online]. Available: https://kubernetes.io/

[5] "Horizontal Pod Autoscaling," Accessed on Dec. 2025, Tech. Rep., section: docs. [Online]. Available: https://kubernetes.io/docs/concepts/workloads/autoscaling/horizontal-pod-autoscale/

[6] I. Alawe, Y. Hadjadj-Aoul, A. Ksentini, P. Bertin, and D. Darche, "On the scalability of 5G core network: The AMF case," in *Proc. of the CCNC*, Jan. 2018, pp. 1–6, iSSN: 2331-9860.

[7] T. V. Kiran Buyakar, H. Agarwal, B. R. Tamma, and A. A. Franklin, "Prototyping and Load Balancing the Service Based Architecture of 5G Core Using NFV," in *Proc. of the IEEE NetSoft*, Jun. 2019, pp. 228–232.

[8] V.-G. Nguyen, K.-J. Grinnemo, J. Taheri, J. Forsman, T. Le Duc, and A. Brunstrom, "On Auto-scaling and Load Balancing for User-plane Gateways in a Softwarized 5G Network," in *Proc. of the CNSM*, Oct. 2021, pp. 132–138, iSSN: 2165-963X.

[9] W. Dev, S. Bera, and A. Tiwari, "Control-Plane Load Balancing and Auto-Scaling in 5G and Beyond Networks," in *Proc. of the International Conference on Information Networking*, Jan. 2025, pp. 83–87.

[10] "Open5GS: Open source 5G core network," Accessed on Jan. 2025. [Online]. Available: https://open5gs.org/

[11] W. Whitt, "APPROXIMATIONS FOR THE GI/G/m QUEUE," *Production and Operations Management*, vol. 2, no. 2, pp. 114–161, 1993.

[12] M. Furqan, Z. Ali, Q. Jan, S. Nazir, S. Iqbal, and Y. Huang, "An Efficient Load-Balancing Scheme for UAVs in 5G Infrastructure," *IEEE Systems Journal*, vol. 17, no. 1, pp. 780–791, Mar. 2023.

[13] A. Chouman, D. M. Manias, and A. Shami, "A Reliable AMF Scaling and Load Balancing Framework for 5G Core Networks," in *Proc. of the International Wireless Communications and Mobile Computing*, Jun. 2023, pp. 252–257.

[14] S. Vittal, S. Sarkar, and A. A. Franklin, "Revamping the Resilience and High Availability of 5G Core for 6G Ready Network Slices," *IEEE Trans. on Network and Service Management*, vol. 21, no. 2, pp. 2287–2302, Apr. 2024.

[15] T.-S.-L. Nguyen, N. Aitsaadi, and C. Adjih, "ANSB: An Optimized Network Slicing Scheme for Adaptive Load Balancing in 5G Core Network," in *Proc. of the ICC*, Jun. 2025, pp. 1402–1407.

[16] A. Khichane, I. Fajjari, N. Aitsaadi, and M. Gueroui, "Cloud Native 5G: An Efficient Orchestration of Cloud Native 5G System," in *Proc. of the IEEE/IFIP NOMS*, Apr. 2022, pp. 1–9.

[17] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, Nov. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8553653

[18] "my5G/my5G-RANTester," my5G Initiative, Tech. Rep., Feb. 2026, original-date: 2020-08-07T17:16:11Z. [Online]. Available: https://github.com/my5G/my5G-RANTester